High-performance VLSI architecture for traffic sign detection

Wang Gangyi[1], Jin Yansheng[2], Ren Guanghui[3], Liu Tong[4]

(1. School of Instrumentation Science and Opto-electronics Engineering, Beihang University, Beijing 100191, China;
2. Jiangsu DFSai Optoelectronic Co., LTD., Nantong 226001, China;
3. School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China;
4. Tianjin Sino-German University of Applied Sciences, Tianjin 300350, China)

Abstract: Traffic sign detection is an important function for driver assistant systems, but the high real-time requirement makes it a very challenging task. A high-performance prohibitory traffic sign detection VLSI (Very Large Scale Integration) architecture was presented. Both color and shape characteristics were utilized in the proposed architecture by detecting circles using circular Hough transform in the red edge bitmap. The local property of circular Hough transform was exploited so that the memory requirement was significantly reduced comparing with common architecture. All the radii were voted concurrently to make full use of the parallelism of logic elements and memory in FPGA. The proposed architecture was implemented on Altera′s EP3C55F484C6 FPGA. The maximum frequency of the implementation was 122 MHz and the resource requirement was acceptable. Experimental results show that the architecture can achieve a throughput up to 115 M pixels/s and was robust to adverse situations such as bad lighting condition, partial occlusion, multiple signs clustered and similar background color.

Key words: traffic sign detection; object detection; circular Hough transform; FPGA; real-time
CLC number: TP391.4 **Document code:** A **DOI:** 10.3788/IRLA201847.0926001

高性能交通标志检测模块的 VLSI 结构设计

王刚毅[1],金炎胜[2],任广辉[3],刘 通[4]

(1. 北京航空航天大学 仪器科学与光电工程学院,北京 100191;
2. 东方赛光电有限公司,江苏 南通 226001;
3. 哈尔滨工业大学 电子与信息工程学院,黑龙江 哈尔滨 150001;
4. 天津中德应用技术大学,天津 300350)

摘 要: 交通标志检测是驾驶辅助系统的重要功能，但对实时性极高的要求使其非常具有挑战性。提出了一种高性能禁令标志检测模块的 VLSI 结构,并在 FPGA 平台上完成了实现和验证。该结构的基本原理是同时利用颜色与形状特征,在图像的红色边缘位图中采用圆霍夫变换检测圆形。通过挖掘圆霍夫变换的局部特性,所提出的结构在内存占用方面显著低于常规结构。所有半径同时投票的设计使 FPGA 的逻辑单元和内存的并行性得以充分发挥。该结构在 Altera 公司的 EP3C55F484C6

收稿日期:2018-04-07； **修订日期:**2018-05-12
基金项目:国家自然科学基金(61605007)
作者简介:王刚毅(1983-),男,讲师,博士,主要从事模式识别、天体敏感器方面的研究。Email: wanggangyi@buaa.edu.cn
通讯作者:金炎胜(1980-),男,工程师,硕士,主要从事图像处理、红外成像、嵌入式系统方面的研究。Email: jannson_3@163.com

型 FPGA 上进行了验证,其最大可运行频率达到 122 MHz,且资源占用在可接受范围内。实验结果表明:该结构的吞吐量达到 115 M 像素/s,且对低光照条件、局部遮挡、多标志相连和相似背景颜色等不利条件具有良好的适应能力。

关键词:交通标志检测; 目标检测; 圆霍夫变换; FPGA; 实时性

0 Introduction

Although traffic signs are designed with fixed colors and shapes, it′s still a challenging computer vision problem to detect traffic signs in complex traffic scenes. The main difficulties include bad lighting condition, similar background color, multiple signs clustered, partial occlusion, etc. In addition, the traffic sign recognition system should detect signs within very short latency so that the driver has enough time to take action.

One important characteristic of traffic signs is that they are painted in fixed colors. Many detection methods extract pixels which have the similar colors as traffic signs in order to quickly get rid of objects of different colors. The major problem for color extraction methods is how to be invariant to different lighting conditions. Some researchers[1] use relative RGB components which are more robust to lighting conditions, some researchers[2] use the hue-saturation-intensity (HSI) color space, which is based on color perception of human beings. The hue and saturation components are invariant to intensity changes, thus color of pixels can be easily determined by thresholding over the hue and saturation components. To reduce the computational cost, some researchers use the hue-saturation-value (HSV) color space, which is similar to HSI, but easier to be converted from the RGB color space[3].

Shape is another important characteristic of traffic signs which are usually circular, rectangular or triangular [4]. Hough transform is a kind of shape detector which can detect such shapes robustly, but the computational complexity and memory consumption is often huge. García et al.[5] use Hough transform in restricted regions to reduce the computational complexity. Barnes et al.[6] propose a Hough like detector based on radial symmetry which can detect regular polygons and circles very fast, but still not in real time. Some other researchers use genetic algorithm to detect shapes. Aoyagi et al.[7] extract points of interest with Laplacian filter to generate a binary image, and search for circles with genetic algorithm. Escalera et al.[8] first extract regions of interest with specific color, and then search for shapes in the parameter space spanned by parameters of translation, rotation and scale. However, the method is too slow for real-time applications.

In this paper, we propose a high-performance VLSI (Very Large Scale Integration) architecture for prohibitory traffic sign detection. We utilize both the color and shape information to detect prohibitory signs which usually have red color and circular shape. The architecture first generates a red bitmap from the input image, and then gets edge pixels and their gradients from the red bitmap. Next, circular Hough transform is performed to detect circles in the edge bitmap, followed by a verification step. Detected circles which pass the verification are finally output as ROIs. We exploit the local property of circular Hough transform, which significantly reduces the memory requirement. The proposed architecture is highly parallelized and requires acceptable logic elements and memory for low-end FPGAs (Field Programmable Gate Array). We implement the architecture on Altera´s EP3C55F484C6 FPGA and get a high throughput up to 115 M pixels/s, which meets the real-time requirements of most

applications. Experimental results show that the architecture is robust to adverse situations such as bad lighting condition, partial occlusion, multiple signs clustered and similar background color.

1 Description of the traffic sign detection method

Before describing the proposed hardware architecture, we first briefly introduce our traffic sign detection method. Figure 1 shows the flow diagram of the method.

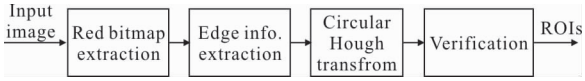


Fig.1 Flow diagram of proposed traffic sign detection method

All the red pixels are first extracted from the input image to generate a red bitmap. To determine whether a pixel is red, HSV color space is employed for its invariance to lighting condition and small computational cost. To convert color space from RGB to HSV, the following equation is used:

$$H=\begin{cases}\left(\dfrac{G-B}{\mathrm{MAX}-\mathrm{MIN}}\bmod 6\right)60^\circ, & \text{if } R=\mathrm{MAX}\\ \left(2+\dfrac{B-R}{\mathrm{MAX}-\mathrm{MIN}}\right)60^\circ, & \text{if } G=\mathrm{MAX}\\ \left(4+\dfrac{R-G}{\mathrm{MAX}-\mathrm{MIN}}\right)60^\circ, & \text{if } B=\mathrm{MAX}\end{cases}$$

$$S=\frac{\mathrm{MAX}-\mathrm{MIN}}{\mathrm{MAX}} \tag{1}$$

$$V=\mathrm{MAX}$$

where MAX and MIN are the maximum and minimum of *RGB* respectively. Then the value of the red bitmap at position x is extracted as follows:

$$RB(x)=\begin{cases}1, & H(x)\geqslant 360^\circ-Th_1, \text{ or } H(x)\leqslant Th_1, \text{ and } S(x)\geqslant Th_2\\ 0, & \text{others}\end{cases} \tag{2}$$

where Th_1, Th_2 are two thresholds for H and S components respectively.

To perform circular Hough transform, edge bitmap E should be extracted from the red bitmap *RB*. Many edge detection methods can be used to extract edge bitmap, but for computational simplicity, we use the following equation to determine whether a pixel is an edge pixel:

$$E=RB-(RB\ominus SE) \tag{3}$$

where SE is a 3-by-3 square kernel, $\ominus$ denotes morphological erosion.

The circular Hough transform also requires the gradient directions of edge pixels. In order to get the gradient direction at edge pixel x, the red degree $f_R(x)$ is first evaluated as follows[9]:

$$f_R(x)=\frac{\min(R(x)-G(x),R(x)-B(x))}{R(x)+G(x)+B(x)} \tag{4}$$

Then the horizontal and vertical gradients $G_x(x)$ and $G_y(x)$ of $f_R(x)$ are calculated with the Sobel operator, and the gradient direction $\theta(x)$ can be calculated as follows:

$$\theta(x)=\arctan\left(\frac{G_y(x)}{G_x(x)}\right) \tag{5}$$

The extracted edge bitmap and gradient directions are sent to the circular Hough transform (CHT) module. We briefly describe circular Hough transform here. Generally, a circle can be represented as the following equation:

$$(x-x_c)^2+(y-y_c)^2=r^2 \tag{6}$$

where (x_c, y_c), r are the center and radius of the circle respectively. Thus, every circle can be mapped to a point in the 3-D parameter space spanned by x_c, y_c, r. To find circles in an edge bitmap, we first vote to the parameter space with the edge pixels. The voting rule is if an edge pixel (x_i, y_i)lies on a circle (x_c, y_c, r), then the circle gets one vote from the point. For the pixel (x_i, y_i) can lie on infinite circles, we need to digitalize the 3-D parameter space into small bins. After voting, we can simply search for the bins with votes more than a threshold, and the parameters of such bins probably correspond to circles in the

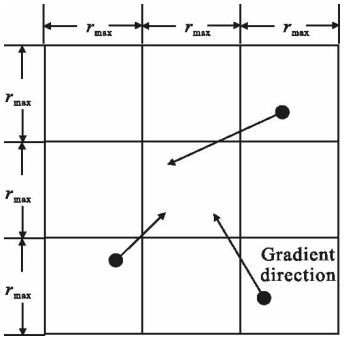
edge bitmap.

It is clear that parameters that voted by an edge pixel generate a 2−D surface in the 3−D parameter space, which makes the computational complexity very large. One way of solving the problem is utilizing the gradient directions of edge pixels. As the gradient direction of any pixel on a circle passes through the center of the circle, we need only vote for the parameters lying on the gradient direction of an edge pixel, which is obviously a 1−D line in the parameter space. Therefore, the computational complexity is reduced significantly.

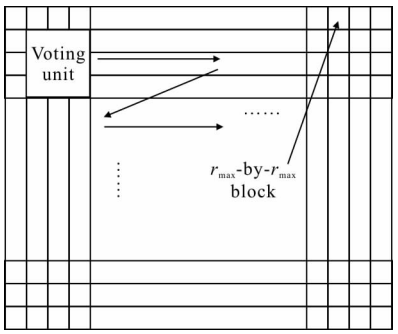
Another problem for circular Hough transform is the huge memory consumption. For a N−by−M image, if we digitalize the parameter space with the precision of 1 pixel, the memory requirement for voting is approximately NMP words, where P is the number of radii to be detected. Such requirement is usually too high for on−chip memory in FPGA, while off−chip memory is also not a good choice for it makes the parallelizing of voting impossible.

We solve this problem by localizing the voting process. In real traffic images, the range of radii of the traffic signs is usually limited. Given that the maximum radius of a traffic sign is $r_{\max}$ in pixels, all the edge pixels (x_i, y_i) voting for a circle centered at (x_c, y_c) are bound to satisfy the conditions: $x_i \in [x_c - r_{\max}, x_c + r_{\max}]$ and $y_i \in [y_c - r_{\max}, y_c + r_{\max}]$. With this property, we design a localized voting unit as shown in Fig.2(a).

The voting unit consists of 9 $r_{\max}$−by−$r_{\max}$ blocks. For circles centered in the central block, all the votes for the circle are bound to come from edge pixels in this voting unit. Therefore, the final voting result in the central block can be obtained without knowing edge information outside the voting unit. To get the voting result of the whole image, we can simply slide the voting unit throughout the image with the step of $r_{\max}$, as shown in Fig.2(b). Therefore, Memory requirement of the whole image is equal to that of the voting unit, which is only $r_{\max}^2 P$, and usually acceptable for on−chip memory in FPGA.



(a) Voting unit



(b) Sliding voting unit throughout the image

Fig.2 Localized voting scheme

After all the candidate circles are detected by the CHT module, the verification module checks whether the circles really exist in the edge bitmap. The criterion is that if more than half of the pixels of a candidate circle are edge pixels, then the candidate circle is determined to be a true circle. Circles satisfying the criterion are finally output as ROIs.

2 Hardware architecture

Based on the method above, we propose an high−performance hardware architecture for detecting prohibitory signs. Figure 3 shows the block diagram of the proposed architecture.

The entire architecture can be divided into

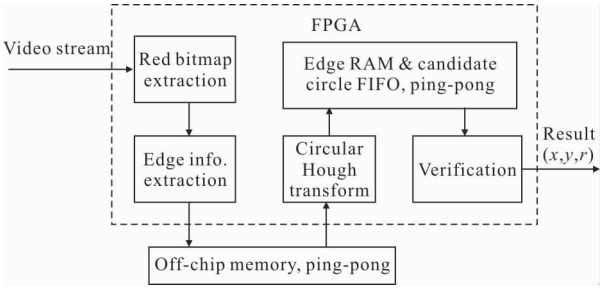


Fig.3 Block diagram of the proposed architecture

two major parts. The preceding part includes the red bitmap extraction module and the edge information extraction module, while the backing part includes the CHT module and the verification module. Edge information is stored in a set of off-chip ping-pong memory which enables the two parts work in parallel. The CHT module and the verification module exchange data with a set of on-chip ping-pong memory named "edge RAM" and "Candidate circle FIFO". We will describe the implementation of each module in the following sections.

2.1 Red bitmap extraction module

The two tasks for the red bitmap extraction module are to determine whether a pixel is red and to calculate the red degree of the pixel. We simplify equation (1) and (4), and design the RTL architecture. The architecture is a 2-stage pipeline, the computational burden is evenly assigned to the two stages to get the maximum throughput. There is a division operation in Equation (4), which could introduce long delay. We divide the operation into two steps: first get the reciprocal of the denominator in a lookup table stored in an on-chip memory, then multiply which with the numerator. As the bit width of the denominator (10 bits) is small, the memory consumption of the lookup table is acceptable.

2.2 Edge information extraction module

As described in Section 2, to get the edge pixels and their gradient directions, the information of the eight neighbors of a n edge pixel is needed. We use two line FIFOs to solve this problem. The red flag and red degree f_R is combined into a word and stored sequentially into the line FIFOs. In each clock cycle, three new words, two from the FIFOs and one from the input port, are pushed into the register bank R1 to R9, thus, the red flags and gradient directions of a 3-by-3 region are stored in R1 to R9. As shown in Fig.4, the directional gradients G_x and G_y are calculated with a 2-stage pipeline and then converted to polar coordinate to get the gradient direction θ. As the values actually needed by the CHT module are the sine and cosine of θ, the two values are directly calculated by changing ρ to 1 and converting (ρ, θ) back to Cartesian coordinate. The edge flag is calculated with a combinational logic operation followed by a series of delay registers to make it synchronized with $\cos\theta$ and $\sin\theta$. Finally, is edge, $\cos\theta$ and $\sin\theta$ are stored together in the off-chip memory.

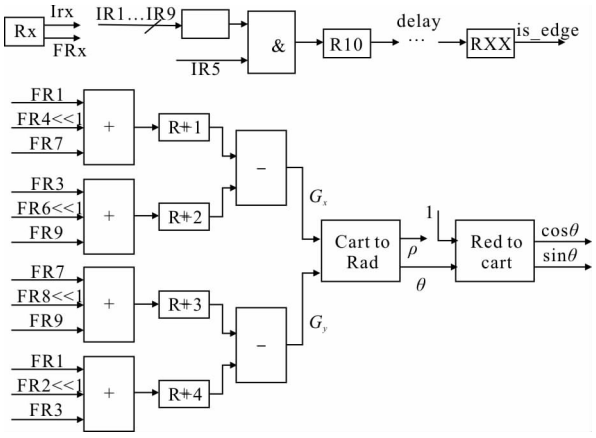


Fig.4 Pipeline for calculating the edge flag and gradient direction

2.3 Circular Hough transform module

After extracting edge information of a frame, the off-chip ping-pong memory is switched, and edge information could be accessed by the circular Hough transform (CHT) module.

The CHT module searches for candidate circles block by block locally, as described in

Section 2. The voting equation is as follows:

$$\begin{cases} x_c = x_i + [r \cdot \cos\theta] \\ y_c = y_i + [r \cdot \sin\theta] \end{cases} \tag{7}$$

where (x_c, y_c), r is the center and radius of the circle, (x_i, y_i) is the edge pixel, $[\cdot]$ means rounding to the nearest integer. If the final votes of a circle are not less than threshold Th_3, then we record (x_c, y_c, r) together with the votes and send them to the verification module.

The architecture of the CHT module is shown in Fig.5. The edge information is first accessed by

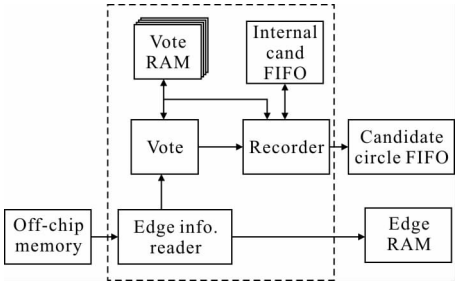


Fig.5 Architecture of the CHT module

the edge information reader submodule, and then sent to the voter submodule. Voter submodule accumulates the votes for each circle which are stored in the vote RAM. While accumulating votes, the voter also monitors all the (x_c, y_c, r) being accumulated, if the votes of any (x_c, y_c, r) equal to the threshold Th_3, the (x_c, y_c, r) are sent to the recorder. In this way, all the candidate circles are recorded during the voting process, and no extra searching process for candidate circles is needed. The recorder stores the candidate circles in the internal candidate FIFO without votes for the final votes are not available during voting. Then after the voting process finished, the recorder reads the final votes of the all the (x_c, y_c, r) stored in the internal candidate FIFO, and output the votes together with the corresponding (x_c, y_c, r) to the external candidate circle FIFO. While sending edge information to the voter submodule, the edge information reader submodule also generates a local edge bitmap within the voting unit, and stores it in the edge RAM, which is used in the verification module.

The voter submodule consists of $P = r_{max} - r_{min} + 1$ sub-voters, each of which corresponds to a radius to be detected. All the sub-voters accumulate votes concurrently, thus, there are also P on-chip RAMs collaborating with the sub-voters. Each RAM has r_{max}^2 words, storing votes of each possible center. The voting process is a 4-stage pipeline, as shown in Fig.6. As we don′t know whether the gradient direction is to or away from the center of the circle, two centers for each radius have to be voted. Therefore, the pipeline is able to process one edge pixel in two clock cycles, unless more than one candidate circles are produced by an edge pixel, in which case the pipeline has to be paused to send the candidates to the recorder submodule in multiple cycles.

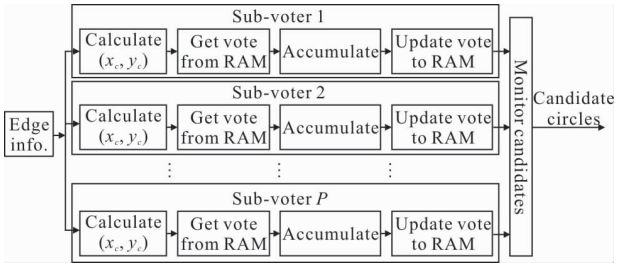
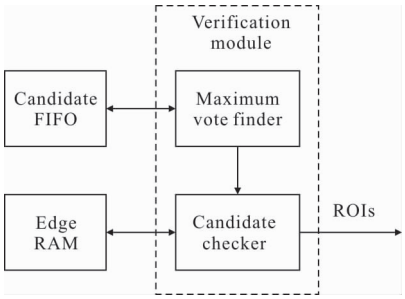


Fig.6 Architecture of the voter submodule

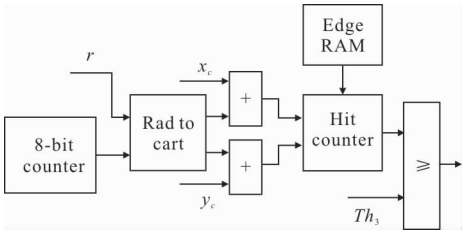
2.4 Verification module

After the CHT module finished voting a block, the ping-pong structure of the edge RAM and candidate circle FIFO are switched so that they can be accessed by the verification module. The task for the verification module is to check whether the candidate circles really exist in the edge bitmap. Figure 7(a) shows the block diagram of this module.

The verification module consists of two submodules. The maximum vote finder submodule searches for the candidate circle with the maximum votes and sends it to the candidate



(a) Entire module



(b) Candidate checker submodule

Fig.7 Architecture of the verification module

checker submodule, which counts the number of edge pixels the circle passes through.

For the maximum vote finder submodule, each (x_c, y_c, r) of the candidate circles are first read from the FIFO, and then compared with the (x_c^*, y_c^*, r^*) of the last verified circle in the block. Verified circle is referred to as a candidate circle which has been proved to be a true circle in this paper. If the parameter (x_{ci}, y_{ci}, r_i) of the i–th candidate circle satisfies $|x_{ci}-x_c^*|\leqslant 7$, $|r_i-r^*|\leqslant 2$ and $|x_{ci}-x_c^*|\leqslant 7$, then the circle is abandoned, which means it is too similar with the verified circle. In this way, we prevent multiple circles are detected from the same circle. The other circles are sent to the vote comparator on the one hand, and on the other hand sent back to the candidate FIFO for the next searching cycle. The vote comparator compares the votes of the input circle with the maximum votes found in current searching cycle, and replaces the maximum votes if the input one is larger. After all the circles in the candidate FIFO have been processed, the circle with the maximum votes is output to the candidate checker submodule, and the vote finder submodule waits until the candidate checker submodule finishes checking the circle, then starts the next searching cycle.

The architecture of the candidate checker submodule is shown in Fig.7 (b). With a polar to Cartesian coordinates convertor, an 8–bit counter and two adders, 256 points evenly distributed on the circle are generated successively. Then the hit counter calculates how many of the points correspond to edge pixels stored in the edge RAM. If the number is not less than threshold Th_4, the circle is verified to be a real circle, and is output as an ROI. On the other hand, (x_c, y_c, r) of the circle is sent back to the maximum vote finder submodule. The searching and checking process continually runs until there is no candidate circle at all in the FIFO.

3 Evaluation

In order to evaluate the proposed architecture, we implement it on FPGA using VHDL language. The selection of parameters, computational accuracy, throughput analysis, and synthesis result are discussed in the following sections.

3.1 Parameter selection

As different results could be produced under different parameters in the architecture, we discuss parameter selection here. Th_1 and Th_2 is used in the red bitmap extraction module to restrict the range of H and S components for a red pixel. According to the experimental results, the detection accuracy is insensitive to slight value changes of the two parameters, therefore, we choose the values of the two parameters empirically. Th_3 used in the CHT module is the minimum vote for a candidate circle. Because too large value for Th_3 can lead to low recall ratio, while too small value increases the burden of the verification module, we choose 9 for a compromise. Th_4 used in the verification module is

the minimum hit of edge pixels for a circle, and we choose 128 to ensure not less than a half of the circle hit edge pixels. r_{min} and r_{max} restrict the range of radii to be detected. As different amount of logic elements (LE) and on-chip memory is needed for different r_{min} and r_{max}, we evaluate two sets of r_{min} and r_{max}.

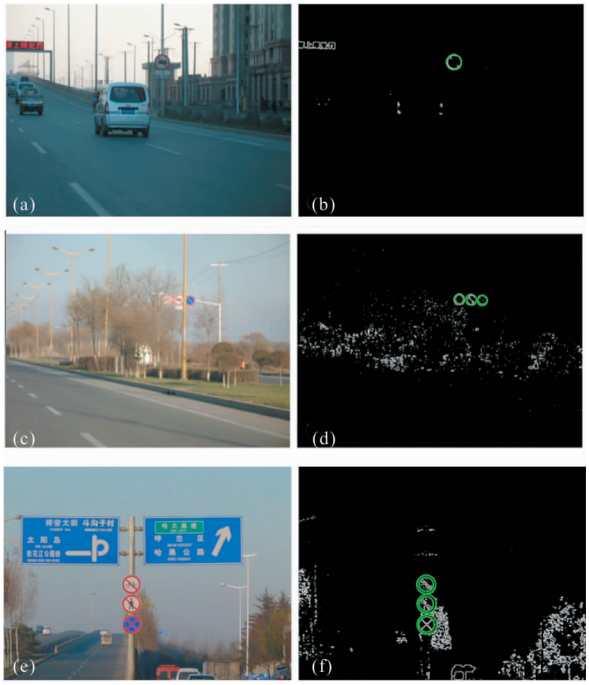
3.2 Simulation and throughput analysis

The proposed architecture is simulated with real images taken in traffic scenes. Results show that the proposed architecture can detect most of the signs and is robust to many kinds of adverse situations. Four typical detection results are shown in Fig.8, which represent adverse situations such as bad lighting condition, partial occlusion, multiple signs clustered and similar background color, respectively.

Table 1 lists the cycle requirement for each module of the architecture to process the images in Fig.8. According to the experiments, the CHT module requires less cycles in most of the images except for the rarely occurred cases in which the number of edge pixels are greater than 4% of the whole image, as shown in Fig.8 (h). Therefore, in most cases, the throughput of the proposed architecture is 0.946 pixels/cycle.

Tab.1 Cycle requirement of the modules in different images

	Edge pixels/all pixels	Red bitmap module	Edge info. module	CHT module	Verification module	Overall
Fig.14(a)	741/307 200	323 968	3 246 670	83 313	1 533	325 336
Fig.14(c)	4 116/307 200	323 968	3 246 670	197 177	2 646	325 336
Fig.14(e)	8 233/307 200	323 968	3 246 670	264 887	3 946	325 336
Fig.14(g)	21 360/307 200	323 968	3 246 670	458 449	1 515	458 984



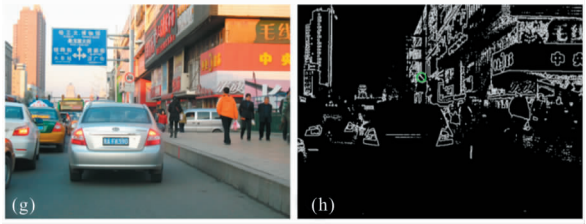


Fig.8 Detection results by the proposed architecture. (a),(c),(e),(g) Original image (b),(d),(f),(h) Edge bitmaps and detected circles

3.3 Synthesis

We synthesize the proposed architecture with Altera Quartus II 11.1 using EP3C55F484C6, and the result is shown in Tab.2. The requirement of logic elements and memory is acceptable, and the throughput of the architecture is up to 115 M pixels/s with the maximum frequency around 120 MHz.

Tab.2 Synthesis result of the proposed architecture

	r_{min}=10, r_{max}=32	r_{min}=10, r_{max}=50
Logic elements	7 464	10 722
Memory/bits	346 496	1 083 596
Maximum frequency/MHz	122	113

The resource requirement can be reduced significantly if we use less voter sub-modules in the CHT module at the price of lower throughput. For video streams with lower pixel rates, such trade-off is worth considering.

4 Conclusion

In this paper, we propose a high performance architecture for prohibitory sign detection. The architecture utilizes both color and shape characteristics of prohibitory signs by detecting circles using circular Hough transform in the red edge bitmap. The local property of circular Hough transform is exploited and the whole image is divided into blocks, in which circles are detected locally. In this way, the computational complexity and the memory requirement of the circular Hough transform is significantly reduced.

The proposed architecture is implemented on Altera′s EP3C55F484C6 FPGA and the resource requirement is acceptable. The maximum frequency is 122 MHz, which enables the architecture to process video streams with pixel rates up to 115 M pixels/s. We evaluate the proposed architecture with images of real traffic scenes, and demonstrate the architecture can robustly detect prohibitory signs under many kinds of adverse situations such as bad lighting condition, partial occlusion, multiple signs clustered and similar background color.

References:

[1] Varan S, Singh S, Sanjeev Kunte R, et al. A road traffic signal recognition system based on template matching employing tree classifier [C]//Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, 2007: 360-365.

[2] Nguwi Y Y, Cho S Y. Emergent self-organizing feature map for recognizing road sign images [J]. ***Neural Computing & Applications***, 2010, 19(4): 601-615.

[3] Garcia -Lamont F, Cervantes J, Lopez A, et al. Segmentation of images by color features: A survey[J]. ***Neurocomputing***, 2018, 292: 1-27.

[4] Hmida R, Ben Abdelali A, Mtibaa A. Hardware implementation and validation of a traffic road sign detection and identification system [J]. ***J Real-Time Image Process***, 2018, 15(1): 13-30.

[5] Garcia-Garrido M A, Sotelo M A, Martin-Gorostiza E. Fast road sign detection using hough transform for assisted driving of road vehicles[C]//Proceedings of the EUROCAST Computer Aided Systems Theory, 2005: 543-548.

[6] Barnes N, Loy G, Shaw D, et al. Regular polygon detection[C]//Proceedings of the Tenth IEEE International Conference on Computer Vision, 2005: 778-785.

[7] Aoyagi Y, Asakura T. A study on traffic sign recognition in scene image using genetic algorithms and neural networks [C]//Proceedings of the IEEE IECON 22nd International Conference on Industrial Electronics, Control, and Instrumentation, 1996: 1838-1843.

[8] Escalera A d l, Armingol J M, Mata M. Traffic sign recognition and analysis for intelligent vehicles [J]. ***Image and Vision Computing***, 2003, 21(3): 247-258.

[9] Ruta A, Li Y M, Liu X H. Real-time traffic sign recognition from video by class-specific discriminative features[J]. ***Pattern Recognition***, 2010, 43(1): 416-430.